RaiBlocks: Una Cryptovaluta con una Rete Distribuita Senza Commissioni

Colin LeMahieu
clemahieu@gmail.com

***Sommario*—Recentemente, L'alta domanda e la limitata scalabilit hanno fatto dilatare i tempi medi delle transazioni e i costi di commissione nelle cryptovalute popolari, rendendo l'esperienza insoddisfacente. A tal proposito introduciamo RaiBlocks, una cryptovaluta con un nuova architettura block-lattice dove ogni account ha la sua propria blockchain, offrendo velocit quasi instantanea per le transazioni e scalabilit illimitata. Ogni utente ha la sua propria blockchain, permettendogli di aggiornarla asincronicamente rispetto al resto della rete, ottenendo come risultato transazioni veloci con il minimo sovraccarico. Viene tenuta traccia del saldo dei conti piuttosto che gli importi delle transazioni, permettendo lo sfoltimento aggressivo del database senza comprometterne la sicurezza. Ad oggi, la rete RaiBlocks ha processato 4.2 milioni di transazioni con un registro alleggerito di soli 1.7GB. L'assenza di commissioni e transazioni in frazioni di secondo fanno di RaiBlocks la miglior cryptomoneta per transazioni di consumo.**

***Termini indice*—cryptovaluta, blockchain, raiblocks, registro distribuito, digitale, transazioni**

I. Introduzione

Dall'implementazione del Bitcoin nel 2009, c' stato un crescente allontanamento dai sistemi finanziari e dalle valute tradizionali sostenute dal governo verso i moderni sistemi di pagamento basati sulla crittografia, che offrono la capacit di conservare e trasferire fondi in modo sicuro e affidabile [1]. Al fine di funzionare efficacemente, una valuta deve essere facilmente trasferibile, non reversibile e deve avere commissioni limitate o assenti. I lunghi tempi delle transazioni, grosse commissioni e scalabilit di rete discutibile hanno sollevato domande sulla pracit del Bitcoin come valuta di tutti i giorni.

In questo documento, introduciamo RaiBlocks, una cryptovaluta a bassa latenza costruita su una struttura dati block-lattice innovativa che offre scalabilit illimitata e transazioni senza commissioni. Il design di RaiBlocks si compone di un protocollo semplice con il solo scopo di essere una cryptovaluta ad elevate prestazioni. Il protocollo di RaiBlocks, infatti, pu essere eseguito su hardware a basso consumo, permettendogli di essere una cryptovaluta pratica, decentralizzata, per l'uso di tutti i giorni.

Le statistiche sulla cryptovaluta riportate in questo documento sono esatte alla data di pubblicazione.

II. Background

Nel 2008, un individuo anonimo sotto lo pseudonimo di Satoshi Nakamoto ha pubblicato un whitepaper illustrando la prima pcryptovaluta decentralizzata al mondo, il Bitcoin [1]. Un'innovazione chiave determinata dal Bitcoin fu la blockchain, una struttura dati pubblica, immutabile e decentralizzata usata come registro per le transazioni della moneta. Sfortunatamente, mentre il Bitcoin maturava, diversi problemi nel protocollo hanno reso il Bitcoin proibitivo per molte applicazioni:

1) Scarsa scalabilit: Ogni blocco nella blockchain puo' memorizzare una quantit limitata di dati, il che significa che il sistema non pu elaborare tantissime transazioni al secondo, rendendo lo spazio utilizzato per il blocco un prodotto di base. Correntemente la commissione minima per una transaziuone e' di $ 10.38 [2].
2) Elevata latenza : Il tempo medio di conferma e' 164 minuti [3].
3) Energeticamente inefficiente: La rete Bitcoin consuma, secondo le stime, 27.28TWh all'anno, usando in media 260KWh per transazione. [4].

Il Bitcoin, e altre cryptomonete, funzionando raggiungendo il consenso sui loro registri globali al fine di verificare le transazioni legittime resistendo ad attori malintenzionati. Il Bitcoin realizza il consenso attraverso una misura economica chiamata Proof of Work (PoW). Nel sistema PoW i partecipanti competono per calcolare un numero, chiamato *nonce*, in modo tale che l'hash dell'intero blocco si trovi in un intervallo di destinazione. Questo valido intervallo e' inversamente proporzionale alla potenza di calcolo cumulativa dell'intera rete Bitcoin al fine di mantenere un tempo medio consistente impiegato per trovare un nonce valido. A colui che trova un nonce valido gli viene permesso di aggiungere il blocco alla blockchain; Di conseguenza, coloro che esauriscono piu' risorse computazionali per calcolare un nonce giocano il ruolo piu' importante nello stato della blockchain. Il PoW fornisce resistenza contro un attacco Sybil, dove un'entita' si comporta come piu' entita' separate per ottenere ulteriore potere in un sistema decentralizzato, riducendo notevolmente le condizioni di concorrenza esistenti intrinsecamente nel momento in cui si accede a una struttura dati globale.

Un protocollo di consenso alternativo, il Proof of Stake (PoS), e' stato introdotto da Peercoin nel 2012 [5]. In un sistema PoS, i partecipanti votano con un peso equivalente alla quantita' di ricchezza che possiedono di una data cryptovaluta. Con questa organizzazione, coloro i quali possiedano un grosso investimento finanziario godono di piu' potere essendo intrinsecamente incentivati a mantenere l'onesta' del sistema altrimenti rischierebbero di perdere il proprio investimento. Il PoS elimina la dispendiosa competizione di calcolo,

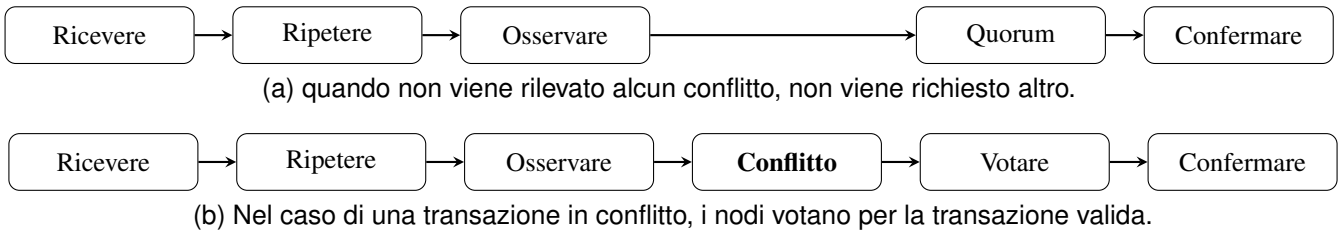

Fig. 1. RaiBlocks non richiede alcun costo aggiuntivo per le transazioni tipiche. Nell'eventualit di transazioni in conflitto, i nodi devonooo e per le transazioni da mantenere

richiedendo soltanto un software leggero in esecuzione su un hardware a basso consumo.

Il paper originare di RaiBlocks e la sua prima implementazione beta furono pubblicati nel dicembre del 2014, rendendo RaiBlocks una delle prime cryptovalute basate sul DAG (Directed Acyclic Graph) [6]. Subito dopo, si e' iniziato a sviluppare altre cryptovalute DAG, in particolare DagCoin/Byteball e IOTA [7], [8]. Queste cryptovalute basate sul DAG hanno "rotto" il vecchio stampo della blockchain, migliorandone le prestazioni di sistema e la sicurezza. Byteball realizza il consenso facendo affidamento su una "main-chain" (chain principale) costituita da "witness" (testimoni) onesti, rispettabili e affidabili, mentre IOTA realizza il consenso tramite il PoW cumulativo di transazioni ammucchiate. RaiBlocks realizza il consenso tramite voto ponderato dal saldo su transazioni in conflitto. Questo sistema di consenso permette transazioni veloci e maggiormente deterministiche ancor mantenendo un sistema forte e decentralizzato. RaiBlocks continuera' su questa linea di sviluppo essendosi posizionata come una delle cryptovalute piu' performanti.

III. Componenti di RaiBlocks

Prima di descrivere l'intera architettura di RaiBlocks, definiamo i componenti individuali che compongono il sistema.

A. Conto

Un account (o conto) e' la parte di chiave pubblica di una coppia di chiavi a firma digitale. la chiave pubblica, chiamata anche indirizzo, e' condivisa con gli altri partecipanti alla rete a differenza della chiave privata che rimane segreta. Un pacchetto di dati digitali assicura che il contenuto sia stato approvato dal titolare della chiave privata. Un utente puo' controllare piu' accounts, ma un solo indirizzo puo' esistere per ogni account.

B. Blocco/Transazione

Il termine "Blocco" e "Transazione" vengono spesso usati intercambiabilmente, dove un blocco contiene una singola transazione. Specificamente, con transazione ci si riferisce all'azione mentre con blocco ci si riferisce alla codifica digitale della transazione. Le transazioni sono firmate dalla chive privata che appartiene al conto sul quele viene eseguita la transazione.

C. Registro

Il registro e' l'insieme globale dei conti in cui ogni conto ha una propria catena di transazioni (Figura 2). Questa e' una componente chiave del design che rientra nella categoria di sostituzione di un accordo di runtime con un accordo in fase di progettazione; tutti concordano attraverso la verifica della firma che solo il proprietario di un account puo' modificare la propria chain. Cio' converte una struttura dati apparentemente condivisa, un libro mastro distribuito, in un insieme di non condivisi.

D. Nodo

Un *nodo* e' un pezzo di software eseguito su un computer conforme al protocollo RaiBlocks e partecipante alla rete RaiBlocks. Il software gestisce il registro ed ogni account che il nodo controlla, se ce ne sono. Un nodo puo' sia memorizzare l'intero registro o una cronologia limitata contenente gli utlimi blocchi della blockchain di ogni account. quando viene installato un nuovo nodo e' raccomandato di verificare l'intera cronologia e sfoltirla localmente.

IV. Panoramica del Sistema

Differentemente dalle blockchain usate in molte altre cryptovalute, RaiBlocks usa una struttura *block-lattice*. Ogni conto ha la sua propria blockchain (account-chain) equivalente alla cronologia transazione/saldo del conto (Figura 2). Ogni

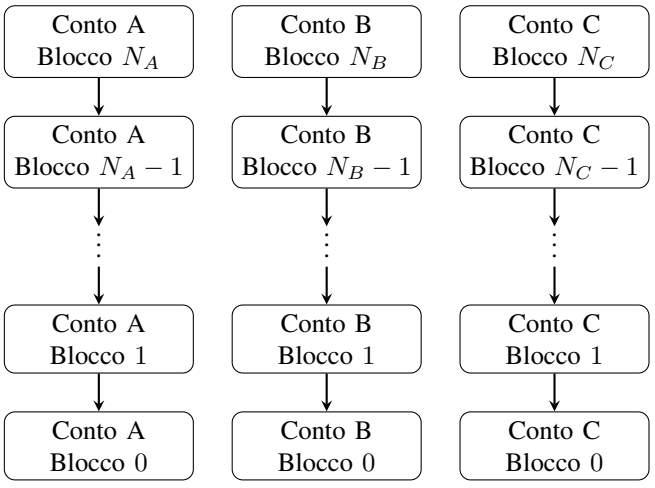

Fig. 2. Ogni conto ha la sua blockchain contenente la cronologia del saldo del conto. Blocco 0 e' una transazione aperta (Sezione IV-B)

account-chain puo' essere aggiornato solamente dal proprietario del conto; cio' permette a ogni account-chain di essere aggiornato immediatamente e asincronicamente rispetto al resto del block-lattice, con il risultato di avere transazioni veloci. Il protocollo di RaiBlocks e' estremamente leggero; ogni transazione calza perfettamente la dimensione minima di un pacchetto UDP per essere trasmesso su internet. anche i requisiti dell'Hardware per i nodi sono minimi, poiche' i nodi devono solo registrare e ritrasmettere i blocchi per la maggior parte delle transazioni (Figura1).

Il sistema e' avviato con un *account genesi* contenente il *saldo genesi*. Il saldo genesi e' una quantita' fissa che non puo' essere aumentata. Il saldo genesi viene diviso e mandato agli altri accounts tramite transazioni registrate sull'account-chain genesi. La somma dei saldi di tutti i conti non eccedera' mai il saldo genesi iniziale che da al sistema un limite superiore sulla quantita', senza possibilita' di essere aumentato.

Questa sezione fara' strada attraverso le differenti tipologie di transazioni costruite e propagate attraverso la rete.

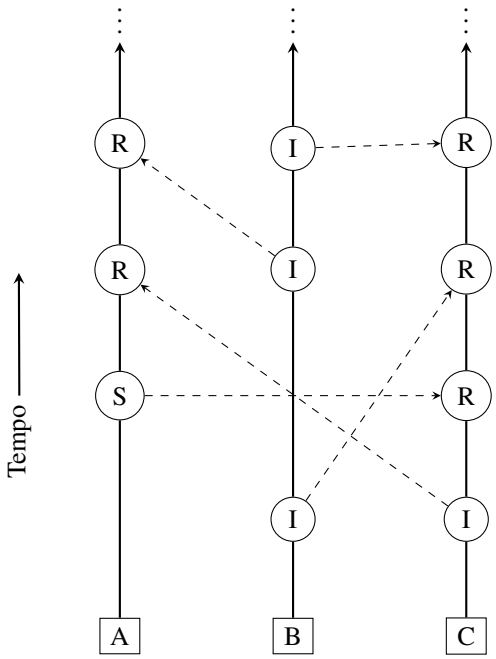

Fig. 3. Visualizzazione del block-lattice. Ogni trasferimento di fondi richiede un blocco inviante (I) e un blocco ricevente (R), ogniuno firmato dal proprietario dell'account-chain (A,B,C)

A. Transazioni

Trasferire fondi da un conto a un altro richiede due transazioni: una di *invio* dedotta dall'ammontare del saldo del mittente e una in *ricezione* che aggiunge l'ammontare al saldo del conto ricevente (Figura 3).

Trasferire fondi come transazioni separate nei conti del mittente e del ricevente serve per realizzare alcuni scopi importanti:

1) Sequenziare i trasferimenti in entrata che sono intrinsecamente asincroni.
2) Mantenere le transazioni in piccole dimensioni per poterle adattare ai pacchetti UDP.
3) Facilitare lo sfoltimento del libro mastro minimizzando il footprint dei dati.
4) Isolare le transazioni regolate da quelle non ancora regolate.

Un'operazione asincrona consiste nell'invio da parte di piu' conti allo stesso conto di destinazione; la latenza di rete e account emittenti che non sono necessariamente in comunicazione tra loro significa che non esiste un modo universalmente accettabile di sapere quale transazione e' avvenuta per prima. Poiche' l'addizione e' associativa, l'ordine con il quale gli inputs sono sequenziati non ha importanza, dunque abbiamo semplicemente bisogno di un accordo globale. Questo e' un componente chive che converte un accordo runtime in un accordo progettato sul tempo. Il conto ricevente ha il controllo sul decidere quale trasferimento sia arrivato per primo e viene espresso dall'ordine firmato dei blocchi in arrivo.

Se un account vuole fare un grosso trasferimento che e' stato ricevuto come insieme di piccoli trasferimenti, vogliamo fare in modo che lo stesso entri in un pacchetto UDP. Quando un conto in ricezione esegue il sequenziamento dei trasferimenti in input, lo stesso mantiene il totale del suo saldo, cosi' che in qualsiasi momento ha la capacita' di trasferire ogni quantita' con una transazione di dimensione fissa. Cio' differisce dal modello di transazione input/output usato dal Bitcoin e altre cryptovalute.

Alcuni nodi non sono interessati nello spendere risorse al fine di memorizzare la cronologia completa di tutte le transazioni; essi sono solamente interessati al saldo corrente di ogni conto. Quando un conto esegue una transazione, esso codifica il suo saldo accumulato e questi nodi hanno solo bisogno di tenere traccia dell'ultimo blocco, il quale gli permette di scartare i vecchi dati senza comprometterne la correttezza.

Anche concentrandoci su accordi basati sul tempo, esiste una finestra di ritardo quando vengono convalidate le transazioni dovuta all'identificazione e alla gestione dei cattivi attori nella rete. siccome gli accordi in RaiBlocks vengono fatti velocemente, nell'ordine di millisecondi, possiamo presentare all'utente due categorie familiari di transazioni in entrata: regolate e non regolate. Le transazioni regolate sono transazioni dove un conto ha generato blocchi in ricezione. Le transazioni non regolate non sono ancora state incorporate nel saldo cumulativo del ricevente. Questo metodo funge da rimpiazzo della piu' complessa e insolita metrica di conferma delle altre cryptomonete.

B. Creare un conto

Per creare un conto, hai bisongo di emettere una transazione *aperta* (Figura 4). Una transazione aperta e' sempre la prima transazione di ongi account-chain e' puo' essere creata sulla prima ricevuta dei fondi. Il campo *conto* conserva la chiave pubblica (indirizzo) derivata dalla chiave privata usata per firmare. Il campo *fonte* contiene l'hash della transazione da dove sono stati mandati i fondi. Alla creazione del conto, occorre scegliere un rappresentante che voti a nome tuo; il quale puo' essere cambiato dopo (Sezione IV-F). Il conto puo' dichiararsi rappresentante di se stesso.

```
open {
   account: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C...182A0E26B4A,
   representative: xrb_1anr...posrs,
   work: 0000000000000000,
   type: open,
   signature: 83B0...006433265C7B204
}
```

Fig. 4. Anatomia di una transazione aperta

C. Saldo del conto

Il saldo del conto viene registrato all'interno del registro stesso. Anziche' registrare la quantita' di una transazione, la verifica (Sezione IV-I) richiede il controllo della differenza tra il saldo del blocco all'invio e il saldo al blocco precedente. Il conto ricevente puo' quindi incrementare il saldo precedente misurato nel saldo finale come indicato nel nuovo blocco in ricezione. Cio' viene fatto per migliorare la velocita' di processamento quando si stanno scaricando blocchi voluminosi. Quando viene richiesta la cronologia del conto, le quantita' sono gia' state date.

D. Invio da un Conto

Per mandare da un indirizzo, l'indirizzo deve necessariamente avere un blocco aperto e pertanto un saldo (Figura 5). Il campo *precedente* contiene l'hash del blocco precedente nell'account-chain. Il campo *destinazione* contiene il conto per i fondi da inviare. Un blocco in invio diviene immutabile una volta confermato. Una volta trasmessi alla rete, i fondi vengono immediatamente dedotti dal saldo del conto emittente e segnalati come *pendenti* fino a che la controparte ricevente non firmi il blocco per accettare detti fondi. I fondi pendenti non devono essere considerati come in attesa di conferma, poiche' sono validi in quanto spesi dal conto emittente in quanto stesso non puo' piu' revocare la transazione.

```
send {
   previous: 1967EA355...F2F3E5BF801,
   balance: 010a8044a0...1d49289d88c,
   destination: xrb_3w...m37goeuufdp,
   work: 0000000000000000,
   type: send,
   signature: 83B0...006433265C7B204
}
```

Fig. 5. Anatomia di una transazione in uscita

E. Ricevere una Transazione

Per completare una transazione, il destinatario dei fondi inviati deve creare un blocco ricevente sul suo account-chain (Figura 6). Il campo sorgente si riferisce all'hash della transazione inviata associata. Una volta che il blocco viene creato e trasmesso, il saldo del conto viene aggiornato e i fondi vengono ufficialmente spostati nel suo conto.

```
receive {
   previous: DC04354B1...AE8FA2661B2,
   source: DC1E2B3F7C6...182A0E26B4A,
   work: 0000000000000000,
   type: receive,
   signature: 83B0...006433265C7B204
}
```

Fig. 6. Anatomia di una transazione in entrata

F. Assegnare un appresentante

L'abilita' dei titolari dei conti di scegliere un rappresentante per votare a loro nome rappresenta un potente strumento di decentralizzazione che non ha forti analogie con i protocolli Proof of Work o Proof of Stake. Nei sistemi PoS convenzionali, il proprietario del conto del nodo deve essere attivo per participare al voto. L'esecuzione continua di un nodo e' scomoda per molti utenti; dando a un rappresentante il potere di votare per tuo conto permette di alleggerire questo requisito. I titolari di un conto hanno l'abilita' di riassegnare il consenso a qualsiasi conto in qualsiasi momento. un *cambio* di transazione cambia il rappresentate di un conto sottraendo il peso del voto dal vecchio rappresentante e aggiungendo il peso del voto a un nuovo rappresentante. (Figura 7). Nessun fondo viene spostato in questa transazione e il rappresentante non ha potere di spesa dei fondi sul conto.

```
change {
   previous: DC04354B1...AE8FA2661B2,
   representative: xrb_1anrz...posrs,
   work: 0000000000000000,
   type: change,
   signature: 83B0...006433265C7B204
}
```

Fig. 7. Anatomia di una transazione in cambiamento

G. Forks e votazioni

Un fork avviene quando j blocchi firmati $b_1, b_2, \ldots, b_j$ rivendicano lo stesso blocco come il loro predecessore (Figura 8). Questi blocchi causano un conflitto sullo stato di un account e pertanto devono essere risolti. Solamente il proprietario del conto ha l'abilita' di firmare i blocchi nel suo account-chain, quindi un fork e' il risultato di una programmazione scadente o di un intento malevolo (double-spend) da parte del proprietario del conto.

Al rilevamento, un rappresentante creera' un voto riferendosi al blocco $\hat{b}_i$ del suo registro trasmettendolo alla rete. Il peso del voto del nodo, w_i, e' la somma dei saldi di tutti i conti che lo hanno nominato come proprio rappresentante. Il nodo osservera' i voti in entrata dagli altri M rappresentanti online e terra' un conteggio cumulato di 4 periodi di votazione, 1 minuto in totale e confermera' il blocco vincente (Equazione 1).

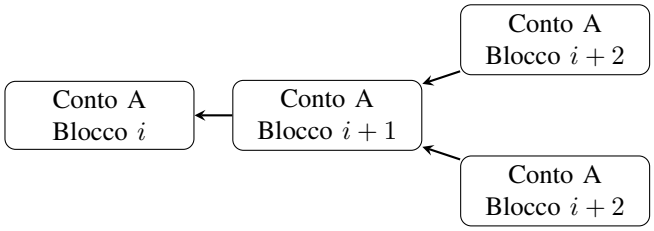

Fig. 8. Un fork accade quando due (o piu') blocchi firmati fanno riferimento allo stesso blocco precedente. Vecchi blocchi non sono a sinistra; Nuovi blocchi non sono a destra

$$v(b_j) = \sum_{i=1}^{M} w_i \mathbb{1}_{\hat{b}_i = b_j} \tag{1}$$

$$b^* = \arg\max_{b_j} v(b_j) \tag{2}$$

Il blocco piu' popolare b^* avra' la maggioranza dei voti e sara' mantenuto nel registro del nodo (Equazione 2). Il blocco(hi) che perde il voto viene scartato. Se un rappresentante rimpiazza un blocco nel suo registro, esso creera' un nuovo voto con una sequenza di numeri maggiore e trasmettera' il nuovo voto alla rete. Questo e' il **solo** scenario dove i rappresentanti votano.

In alcune circostanze, brevi problemi di connettivita' di rete potrebbero causare la non accettazione di un blocco trasmesso da parte di tutti i peers. Qualsiasi blocco successivo su questo conto sara' ignorato perche' invalidato dai peers che non hanno visto la sua trasmissione iniziale. Una ritrasmissione di questo blocco sara' accettata dai peers rimanenti e i blocchi successivi saranno recuperati automaticamente. Anche quando avviene un fork o manca un blocco, soltanto gli account di riferimento nella transazione ne sono affetti; Il resto della rete procede con il processamento delle transazioni per tutti gli altri conti.

H. Proof of Work

Tutti e quattro i tipi di transazione hanno un campo di lavoro che deve essere compilato correttamente. Il campo di lavoro permette al creatore della transazione di calcolare un nonce in modo tale che l'hash concatenato con il nonce del campo precedente in transazioni ricevere/mandare/cambiare o il campo del conto in una transazione aperta sia al di sotto di un certo valore di soglia. Diversamente dal Bitcoin, il PoW in RaiBlocks e' semplicemente usato come tool antispam, similarmente ad Hashcash e pu essere calcolato nell'ordine di secondi [9]. Una volta che una transazione e' inviata, il PoW per il successivo blocco puo' essere precalcolato fintanto che il campo del blocco precedente e' conosciuto; Cio' permette di far apparire le transazioni instantaneamente a un utente finale fintanto che il tempo tra le transazioni e' maggiore del tempo richiesto per calcolare il PoW.

I. Verificazione delle Transazioni

Un blocco per essere considerato valido, deve avere i seguenti attributi:

1) Il blocco non deve essere gia' nel libro mastro (duplicate transaction).
2) Deve essere firmato dal proprietario del conto.
3) Il blocco precedente e' la testa dell'account-chain. Se esso esiste ma non e' la testa, e'un fork.
4) Il conto deve avere un blocco aperto.
5) l'hash calcolato deve soddisfare i requisiti di soglia PoW.

Se e' un blocco in arrivo, controlla che sia pendente l'hash del blocco sorgente, significa che non e' stato ancora riscattato. Se si tratta di un blocco di invio, il saldo deve essere inferiore al saldo precedente.

V. Vettori di Attacco

RaiBlocks, come tutte le valute decentralizzate, puo' essere attaccato da parti malintenzionate per tentato guadagno finanziario o cessazione del sistema. in questa sezione verranno delineati alcuni scanari di attacchi possibili, le conseguenze di tali possibili attacchi e come il protocollo RaiBlocks prenda misure preventive.

A. Divario di Sincorinizzazione del Blocco

Nella senzione IV-G, abbiamo discusso lo scenario dove un blocco potrebbe non essere propriamente trasmesso, causando l'ignoramento dei blocchi successivi da parte della rete. se un nodo riceve un blocco che non abbia il riferimento al blocco precedente, ha due opzioni:

1) Ignorare il blocco in quanto potrebbe essere un blocco inutile e dannoso.
2) richiedere una risincronizzazione con un altro nodo.

Nel caso di una risincronizzazione, una connessione TCP deve essere formata con un nodo bootstrap per facilitare l'aumento della quantita' di traffico richiesto da una risincronizzazione. comunque, se il blocco era in realta' un blocco errato, la risincronizzazione non e' necessaria e ha solo aumentato inutilmente il traffico sulla rete. Si tratta di un Network Amplification Attack e risulta in un Denial-of-Service (DoS).

Per evitare una risincronizzazione non necessaria, i nodi aspetteranno fin quando una certa soglia di voti non sia stata osservata per un blocco potenzialmente melevolo prima di iniziare una connessione bootstrap a un nodo per risincronizzare. se un blocco non riceve abbastanza voti puo' essere considerato spazzatura.

B. Inondazione di Transazioni

Un'entita' malevola puo' inviare molte transazioni valide ma non necessarie fra conti sotto il suo controllo nel tentativo di saturare la rete. Senza nessuna commissione per le transazioni la stessa sarebbe capace di continuare questo attacco indefinitivamente. Comunque, Il PoW richiesto per ogni transazione limita la percentuale di transazioni che l'entita' malevola potrebbe generare senza investire in risorse computazionali. Anche nel caso di un tale attacco nel tentativo di inflazionare il libro mastro, i nodi che non possiedono l'intera cronologia sono capaci di sfoltire le vecchie transazioni dalla loro chain; questo restringe l'utilizzo di memoria da questo tipo di attacco per quasi tutti gli utenti.

C. Attacco Sybil

Un'entita' potrebbe creare centinaia di nodi Raiblocks su una sola macchina; comunque, siccome il sistema di voto e' ponderato sulla base del saldo del conto, aggiungere altri nodi alla rete non consentira' all'attaccante di guadagnare voti extra. Pertanto non vi e' alcun vantaggio da guadagnare con un attacco Sybil.

D. Attacco Penny-Spend

Questo attacco consiste nello spendere quantita' infinitesimali su un gran numero di conti al fine di esaurire le risorse di memoria dei nodi. Il tasso di pubblicazione dei blocchi e' limitata dal PoW, cio' limita la creazione di conti e transazioni a una certa entita'. I nodi che non possiedono l'intera cronologia possono sfoltire i conti sotto una metrica statistica quando il conto risulta molto propbabilmente invalido. Infine, RaiBlocks e' progettato per utilizzare una minima quantita' di memoria, quindi lo spazio richiesto per memorizzare un conto addizionale e' proporzionale alla dimensione di un open block + indexing $= 96\text{B} + 32\text{B} = 128\text{B}$. Cio' equivale a 1GB, capace di memorizzare 8 milioni di account penny-spend. Se i nodi vogliono fare uno sfoltimento piu' aggressivo, possono calcolare la distribuzione basata sulla frequenza di accesso e i conti delegatari usati di rado dall'archiviazione piu' lenta.

E. Attacco PoW Precalcolato

Dal momento che il proprietario di un conto sara' l'unica entita' ad aggiungere blocchi all'account-chain, possono essere computati blocchi in sequenza, insieme al loro PoW, prima di essere trasmessi alla rete. L'attaccante genera una miriade di blocchi in sequenza, ogniuno con un valore minimo, su un lungo periodo di tempo. A un certo punto, l'attaccante performa una negazione del servizio (DoS) inondando la rete con un sacco di transazioni valide, le quali vengono processate dagli altri nodi che ne fanno eco il piu' velocemente possibile. Questa e' una versione avanzata dell'inondazione di transazioni descritta nella sezione V-B. Un tale attacco funzionerebbe solo brevemente, ma potrebbe essere usato insieme ad altri attacchi, come >50% attack (Sezione V-F) per aumentarne l'efficacia. Limitazione della percentuale di transazioni e altre tecniche vengono correntemente studiare per mitigare questo tipo di attacchi.

F. >50% Attack

La metrica di consenso per RaiBlocks e' un sistema di voto ponderato in base al saldo. se un attaccante e' in grado di guadagnare piu' del 50% della forza voto, puo' influenzare il consenso della rete rendendo il sistema "guasto". Un utente malintenzionato in grado di ridurre la quantit di saldo che deve perdere impedendo ai nodi buoni di votare attraverso un DoS di rete. RaiBlocks adotta le seguenti misure per prevenire un simile attacco:

1) La prima difesa contro questo tipo di attacco e' una votazione ponderata legata all'investimento nel sistema. Il proprietario di un conto e' intrinsecamente incentivato a mantenere l'onesta' del sistema stesso per proteggere il suo investimento. Il tentativo di capovolgere il libro mastro sarebbe distruttivo per l'intero sistema portando alla distruzione del loro investimento.
2) Il costo di questo attacco e' proporzionale alla capitalizzazione di mercato di RaiBlocks. nel sistema PoW, puo' essere inventata una tecnologia che dia un controllo sproporzionato comparato all'investimento monetario e nel caso in cui l'attacco abbia successo, detta tecnologia puo' essere reimpiegata dopo che l'attaco viene portato a termine. In RaiBlocks il costo per attaccare il sistema e' scalabile con il sistema stesso, quindi se un attaco risulterebbe possibile l'investimento utilizzato per l'attacco, al contrario del PoW, non e' recuperabile o riutilizzabile.
3) Al fine di mantenere il quorum massimo di voti, la prossima linea di difesa il voto rappresentativo. I proprietari dei conti che non sono in grado di partecipare in maniera affidabile alla votazione per problemi di connettivita' possono nominare un rappresentante che voti con il peso del loro saldo. Massimizzando il numero e la diversita' dei rappresentanti aumenta la resilienza della rete.
4) I forks in RaiBlocks non avvengono mai accidentalmente, quindi i nodi possono prendere decisioni politiche su come interagire con i blocchi forkati. L'unico caso in cui gli account dei non attaccanti sono vulnerabili ai blocchi forkati quando ricevono un saldo da un account in attacco. Gli accounts che desiderano essere protetti da forks possono aspettare un po' o molto tempo prima di ricevere il blocco da un conto che ha generato il fork oppure optare per non riceverne mai. Coloro che ricevono possono anche generare accounts separati da usare per quando ricevono dei fondi da conti di dubbia provenienza al fine di isolare gli altri accounts.
5) Una linea di difesa finale non ancora implementata e' il *block cementing*. RaiBlocks fa tutto il possibile per sistemare rapidamente i forks tramite voto. I nodi potrebbero essere configurati per "cementare" i blocchi, impedendo un ritorno allo stato precedente dopo un certo periodo di tempo. In questo modo la rete e' sufficientemente resa sicura, concentrandosi su tempi di assestamento rapidi per evitare forks ambigui.

Una versione piu' sofisticata dell'attacco $> 50\%$ e' descritto in figura 9. "Disconnesso" e' la percentuale di rappresentati nominati che non sono online per votare. "Stake" e' la quantita' di investimento con cui l'utente malevolo sta votando. "Attivo" sono i rappresentanti online che stanno votando secondo protocollo. Un attaccante puo' compensare l'ammontare di quota che deve perdere mandando offline altri votanti tramite un attacco DoS di rete. Se viene sostenuto questo attacco, i rappresentanti attaccati verranno disincronizzati e cio' viene mostrato da "Non Sincronizzato." Infine, un attaccante puo' ottenere una breve scarica nella forza di voto relativo scambiando il suo attacco di tipo Denial of Service su un nuovo gruppo di rappresentanti mentre il vecchio set sta ancora risincronizzando il libro mastro, mostrato da "Attacco."

se un attaccante e' capace di causare uno Stake >Active

Disconnesso	NonSincronizzato	**Attacco**	Attivo	Stake

Fig. 9. Un potenziale accordo di voto che potrebbe abbassare il 51% come requisito di attacco.

tramite una combinazione di queste circostanze, sarebbe in grado di capovolgere con successo i voti sul registro a scapito del loro stake. Possiamo stimare quanto potrebbe costare questo tipo di attacco esaminando la capitalizzazione di mercato di altri sistemi. Se stimiamo che il 33% dei rappresentanti sono offline o attaccati tramite un DoS, un attaccante avrebbe bisogno di comprare il 33% della capitalizzazione di mercato al fine di attacare il sistema tramite votazione.

G. Avvelenamento da Bootstrap

Piu' a lungo un utente malintenzionato riesce a mantenere una vecchia chiave privata con saldo, maggiore e' la probabilita' che i saldi esistenti a quel momento non abbiamo rappresentanti che vi partecipino perche' i loro saldi o rappresentanti sono stati trasferiti a conti piu' recenti. cio' significa che se un nodo viene riavviato su una vecchia rappresentazione della rete dove l'attacante ha un quorum di quota di voto rispetto ai rappresentanti in quel momento, sarebbe in grado di influenzare le decisioni di voto su quel nodo. se questo nuovo utente volesse interagire con qualcuno oltre al nodo attaccante, tutte le sue transazioni verrebbero negate poiche' esse hanno blocchi testa diverse. Il risultato e' che i nodi possono sprecare il tempo di nuovi nodi nella rete inviandogli cattive informazioni. Per prevenire cio', i nodi possono essere accoppiati con un database iniziale di conti e blocchi testa buoni conosciuti; cio' e' un rimpiazzo del download del database fino al blocco genesi. Quanto piu' vicino e' il download dall'essere attuale, maggiore la probabilit di difendersi con precisione contro questo attacco. Alla fine, questo attacco non probabilmente peggiore di quello di inviare dati inutili ai nodi durante l'avvio automatico, dal momento che non sarebbero in grado di effettuare transazioni con chiunque abbia un database contemporaneo.

VI. Implementazione

Attualmente l'implementazione di riferimento implementata in C++ e sta producendo versioni dal 2014 su Github [10].

A. Caratteristiche del progetto

L'implementazione di RaiBlocks aderisce allo standard di architettura delineato in questo documento. Ulteriori specifiche sono descritte qui.

1) Algoritmo di firma: RaiBlocks utilizza un algoritmo di curva ellittica ED25519 modificato con hashing Blake2b per tutte le firme digitali [11]. ED25519 stato scelto per la firma rapida, la verifica rapida e l'alta sicurezza.

2) Algoritmo di Hashing: Poich l'algoritmo di hashing viene utilizzato solo per prevenire lo spam di rete, la scelta dell'algoritmo meno importante rispetto alle criptovalute basate sul mining. La nostra implementazione utilizza Blake2b come algoritmo di digest contro i contenuti del blocco [12].

3) Funzione di Derivazione Chiave: Nel wallet di riferimento, le chiavi sono crittografate da una password e la password viene fornita tramite una funzione di derivazione della chiave per proteggerla dai tentativi di cracking ASIC. Attualmente Argon2 [13] il vincitore dell'unica competizione pubblica volta a creare una funzione di derivazione chiave resiliente.

4) Intervallo Blocco: Dato che ogni conto ha la propria blockchain, gli aggiornamenti possono essere eseguiti in modo asincrono rispetto allo stato della rete. Pertanto non ci sono intervalli di blocchi e le transazioni possono essere pubblicate istantaneamente.

5) Protocollo messaggio UDP: Il nostro sistema e' progettato per operare indefinitamente usando la minima quantita' minima di risorse di calcolo possibile. Tutti i messaggi nel sistema sono stati progettati per essere senza stato e calzare un singolo pacchetto UDP. Cio' rende anche semplice, per i peer leggeri con connettivita' intermittente, di partecipare alla rete senza ristabilire connessioni TCP a breve termine. Il TCP e' usato solamente per i nuovi peers quando essi vogliono avviare la blockchain in maniera massiccia.

I nodi possono essere sicuri che le loro transazioni siano state ricevute dalla rete osservando il traffico delle transazioni trasmesse dagli altri nodi come si dovrebbe vedere che differenti copie fanno eco a se stesse.

B. IPv6 e Multicast

Costruire su UDP senza connessione permette future implementazioni per usare IPv6 Multicast come sostituto delle tradizionali inondazioni di transazioni e trasmissione del voto. Questo ridurra' il consumo di larghezza di banda e dara' piu' flessibilita' politica ai nodi andando avanti.

C. Prestazione

Al tempo della stesura di questo documento, 4.2 milioni di transazioni sono state processate dalla rete RaiBlocks, ottenendo una dimensione della blockchain di 1.7GB. I tempi di transazione sono misurati nell'ordine di secondi. Un'attuale implementazione di riferimento che operata su unita' SSD puo' elaborare oltre 10,000 transazioni al secondo principalmente in termini IO.

VII. Utilizzo delle Risorse

Questa e' una panoramica delle risorse usate da un nodo RaiBlocks. Inoltre, esaminiamo nuove idee per ridurre l'utilizzo delle risorse in casi d'uso specifici. Nodi ridotti sono tipicamente chiamati leggeri, sfoltiti o nodi SPV (Simplified Payment Verification).

A. Rete

La quantita' di attivita' della rete dipende da quanto la rete contribuisce verso la salute stessa di una rete.

1) Rappresentante: Un nodo rappresentante richiede massime risorse di rete siccome esso osserva il traffico di voto di altri rappresentanti e pubblica i propri voti.

2) Affidabile: Un nodo affidabile e' simile a un nodo rappresentante ma e' solo un osservatore, non ha alcuna chiave privata rappresentativa di un conto e non pubblica voti propri.

3) Fidato: Un nodo fidato osserva il traffico di voto da un rappresentante di cui si fida per eseguire correttamente il consenso. Questo riduce la quantita' di traffico di voti in entrata da parte dei rappresentanti che si recano su questo nodo.

4) Leggero: Un nodo leggere e' anche un nodo affidabile che osserva solamente il traffico per i conti ai quali e' interessato permettendo un uso minimo della rete.

5) Bootstrap: Un nodo bootstrap serve parti o l'intero libro mastro ai nodi che si mettono online. cio' viene fatto su una connessione TCP invece della connessione UDP siccome implica una grande quantita' di dati che richiede un controllo avanzato del flusso di rete.

B. Capacita' del Disco

Dipendendo dalla domanda degli utenti, diverse configurazioni dei nodi richiedono requisiti di archiviazione diversi.

1) Cronologia: Un nodo interessato a conservare l'intera cronologia di tutte le transazioni avra' bisogno della quantita' massima di spazio di archiviazione.

2) Corrente: A causa della progettazione di mantenere i saldi accumulati con i blocchi, i nodi hanno solo bisongo di mantenere l'ultimo o i blocchi di testa per ogni conto per poter partecipare al consenso. se un nodo non e' interessato a tenere la cronologia completa puo' optare di tenere solo i blocchi di testa.

3) Leggero: Un nodo leggero non mantine alcun dato sul registro locale e inoltre partecipa alla rete solamente per osservare l'attivita' dei conti nei quali egli e' interessato o opzionalmente creare nuove transazioni con le chiavi private che possiede.

C. CPU

1) Generazione di Transazioni: Un nodo interessato nel creare nuove transazioni deve produrre un nonce Proof of Work al fine di passare il meccanismo di throttling di RaiBlocks. Il calcolo di vari componenti hardware analizzato nell'appendice A.

2) Rappresentante: Un rappresentante deve verificare le firme dei blocchi, dei voti e anche produrre la propria firma per partecipare al consenso. La quantita' di risorse CPU per un nodo rappresentante e' significativamente piu' bassa della generazione di transazioni e dovrebbe funzionare con qualsiasi CPU dei moderni computer.

3) Osservatore: Un nodo osservatore non genera alcun voto proprio. siccome la generazione di firme e' minima, i requisiti della CPU sono quasi identici a quelli quando si esegue un nodo rappresentante.

VIII. Conclusione

In questo documento e' stata presentata l'architettura di una cryptovaluta affidabile, senza commissioni, a bassa latenza che utilizza un nuova struttura block-lattice e un nuovo meccanismo Proof of stake di votazione delegato. La rete richiede le minime risorse, nessun hardware da mining di grossa potenza, e puo' processare elevatissime transazioni. Tutto cio' viene reso possibile avendo blockchain individuali per ogni conto, eliminato problemi di accesso e inefficienze di una struttura dati globale. Abbiamo identificato possibili vettori di attacco al sistema e presentato argomentazioni su come RaiBlocks e' resistente a queste forme di attacco.

Appendice A
Benchmarks PoW Hardware

Come menzionato precedentemente, lo scopo del PoW in RaiBlocks e' quello di ridurre lo spam nella rete. L'implementazione del nostro nodo fornisce un acceleratore che trae vantaggio dalle GPU compatibili con OpenCL. La tavola I fornisce un confronto reale fra i vari tipi di hardware. Correntemente la soglia di PoW e' fissa, ma una soglia adattabile potrebbe essere implementata man mano che la potenza di calcolo media aumenta.

TAVOLA I
Prestazioni Hardware PoW

Dispositivo	Transazioni per Secondo
Nvidia Tesla V100 (AWS)	6.4
Nvidia Tesla P100 (Google,Cloud)	4.9
Nvidia Tesla K80 (Google,Cloud)	1.64
AMD RX 470 OC	1.59
Nvidia GTX 1060 3GB	1.25
Intel Core i7 4790K AVX2	0.33
Intel Core i7 4790K,WebAssembly (Firefox)	0.14
Google Cloud 4 vCores	0.14-0.16
ARM64 server 4 cores (Scaleway)	0.05-0.07

Riconoscimenti

Ci piacerebbe ringraziare Brian Pugh per aver compilato e formattato questo documento.

Riferimenti

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: http://bitcoin.org/bitcoin.pdf
[2] "Bitcoin median transaction fee historical chart." [Online]. Available: https://bitinfocharts.com/comparison/bitcoin-median_transaction_fee.html
[3] "Bitcoin average confirmation time." [Online]. Available: https://blockchain.info/charts/avg-confirmation-time
[4] "Bitcoin energy consumption index." [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption
[5] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf
[6] C. LeMahieu, "Raiblocks distributed ledger network," 2014.
[7] Y. Ribero and D. Raissar, "Dagcoin whitepaper," 2015.
[8] S. Popov, "The tangle," 2016.
[9] A. Back, "Hashcash - a denial of service counter-measure," 2002. [Online]. Available: http://www.hashcash.org/papers/hashcash.pdf
[10] C. LeMahieu, "Raiblocks," 2014. [Online]. Available: https://github.com/clemahieu/raiblocks
[11] D. J. Bernstein, N. Duif, T. Lange, P. Shwabe, and B.-Y. Yang, "High-speed high-security signatures," 2011. [Online]. Available: http://ed25519.cr.yp.to/ed25519-20110926.pdf
[12] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein, "Blake2: Simpler, smaller, fast as md5," 2012. [Online]. Available: https://blake2.net/blake2.pdf
[13] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: The memory-hard function for password hashing and other applications," 2015. [Online]. Available: https://password-hashing.net/argon2-specs.pdf